Methods for Efficient Unfolding of Colored Petri Nets

Alexander Bilgram, Peter G. Jensen, Thomas Pedersen,
Jiří Srba*, **Peter H. Taankvist**

Aalborg University

Department of Computer Science, Aalborg, Denmark

pgj@cs.aau.dk, srba@cs.aau.dk

Abstract. Colored Petri nets offer a compact and user friendly representation of the traditional Place/Transition (P/T) nets and colored nets with finite color ranges can be unfolded into the underlying P/T nets, however, at the expense of an exponential explosion in size. We present two novel techniques based on static analysis in order to reduce the size of unfolded colored nets. The first method identifies colors that behave equivalently and groups them into equivalence classes, potentially reducing the number of used colors. The second method overapproximates the sets of colors that can appear in places and excludes colors that can never be present in a given place. Both methods are complementary and the combined approach allows us to significantly reduce the size of multiple colored Petri nets from the Model Checking Contest benchmark. We compare the performance of our unfolder with state-of-the-art techniques implemented in the tools MCC, Spike and ITS-Tools, and while our approach is competitive w.r.t. unfolding time, it also outperforms the existing approaches both in the size of unfolded nets as well as in the number of answered model checking queries from the 2021 Model Checking Contest.

1. Introduction

Petri nets [1], also known as P/T nets, are a powerful modelling formalism supported by a rich family of verification techniques [2]. However, P/T nets often become too large and incomprehensible for humans to read. Therefore, colored Petri nets (CPN) [3] were introduced to allow for high level modelling of distributed systems. In CPNs, each place is assigned a color domain and each token in that place has a color from its domain. Arcs have expressions that define what colored tokens to consume or produce, and transitions have guard expressions that restrict transition enabledness.

---

*Address for correspondence: Aalborg University, Department of Computer Science, Aalborg, Denmark.

A CPN can be translated into an equivalent P/T net, provided that every color domain is finite, through a process called *unfolding*. This allows us to use efficient verification tools already developed for P/T nets. When unfolding a CPN, each place is unfolded into a new place for each color that a token can take in that place; a naive approach is to create a new place for each color in the color domain of the place. Transitions are unfolded such that each binding of variables to colors, satisfying the guard, is unfolded into a new transition copy in the unfolded net. The size of an unfolded net can be exponentially larger than the colored net and the unfolding process therefore requires optimizations in order to finish in realistic time and memory. Several types of improvements were proposed that analyse transition guards and arc expressions [4, 5, 6]. However, even with these optimizations, there still exist CPNs that cannot be unfolded using the existing tools. As an example, the largest instances of the nets *FamilyReunion* [7, 8] and *DrinkVendingMachine* [9, 10] from the Model Checking Contest [11] have not yet been unfolded in the competition setup.

We propose two novel methods for statically analysing a CPN to reduce the size of the unfolded P/T net. The first method called *color quotienting* uses the fact that sometimes tokens with different colors can be indistinguishable in the sense that they generate bisimilar behaviour. If such colors exist in the net, we can create equivalence classes that represent the colors with similar behaviour. As such, we can reduce the amount of colors that we need to consider when unfolding. The second method called *color approximation* overapproximates which colors can possibly be present in any given place such that we only unfold places for the colors that can exist. This method also allows for invalidating bindings that are dependent on unreachable colors, thus reducing the amount of transitions that are unfolded.

Our two methods are implemented in the model checker TAPAAL [12, 13] and an extensive experimental evaluation shows convincing performance compared to the state-of-the-art tools for CPN unfolding.

Related work.   Heiner et al. [5] analyse the arc and guard expressions to reduce the amount of bindings by collecting *patterns*. The pattern analysis is implemented in the tool Snoopy [14] and our color approximation method further extends this method. In [6] the same authors present a technique for representing the patterns as Interval Decision Diagrams. This technique is used in the tools Snoopy [14], MARCIE [15] and Spike [16] and performs better compared to [5]; it also allows to unfold a superset of colored nets compared to the format adopted by the Model Checking Contest benchmark [11].

In [4] Dal-Zilio describes a method (part of the unfolder MCC) called *stable places*. A stable place is a place that never changes from the initial marking, i.e. every time a token is consumed from this place an equivalent token of the same color is added to the place. This method is especially efficient on the net BART from the Model Checking Contest [11], however, it does not detect places that deviate even a little from the initial marking. Our color approximation method includes a more general form of the stable places. In the unfolder MCC [4], a *component analysis* is introduced and it detects if a net consists of a number of copies of the same component. MCC is used in the TINA toolchain [17] and to our knowledge in the latest release of the LoLA tool [18]. GreatSPN [19] is another tool for unfolding CPNs, however, in [4] it is demonstrated that MCC is able to greatly outperform GreatSPN and as such we omit GreatSPN from later experiments.

ITS-Tools [20] has an integrated unfolding engine. The tool uses a technique of *variable symmetry identification*, in which it is analyzed whether variables x and y are permutable in a binding. Furthermore, they use stable places during the binding and they apply analysis to choose the binding order of parameters to simplify false guards as soon as possible. After unfolding, ITS-Tools applies further post-unfolding reductions that remove orphan places/transitions and behaviourally equivalent transitions [21]. Our implementation includes a variant of the symmetric variables reduction as well. In [22] Thierry-Mieg et al. present a technique for automatic detection of symmetries in high level Petri nets used to construct symbolic reachability graphs in the GreatSPN tool. This detection of symmetries is reminiscent of the color quotienting method presented in this paper, although our color quotienting method is used for unfolding the colored Petri net instead of symbolic model checking.

In [23] Klostergaard presents a simple unfolding method implemented in TAPAAL [12, 13], which is the base of our implementation. The implementation is efficient but there are several nets which it cannot unfold. Both unfolding methods introduced in this paper are advanced static analyses techniques and we observe that the above mentioned techniques, except symmetric variables and component analysis, are captured by color approximation and/or color quotienting.

This paper is an extended version of the conference paper [24] with full proofs, complete definitions and additional examples and last but not least a substantial reimplementation of the methods in the tool TAPAAL with improved experimental results compared to [24].

2. Preliminaries

Let $\mathbb{N}^{>0}$ be the set of positive integers and $\mathbb{N}^0$ the set of nonnegative integers. A Labeled Transition System (LTS) is a triple $(Q, Act, \rightarrow)$ where Q is a set of states, Act is a finite, nonempty set of actions, and $\rightarrow \subseteq Q \times Act \times Q$ is the transition relation. We write $s \xrightarrow{a}$ if there is $s' \in Q$ such that $s \xrightarrow{a} s'$ and $s \not\xrightarrow{a}$ if there is no such s'. A binary relation R over the set of states of an LTS is a *bisimulation* iff for every $(s_1, s_2) \in R$ and $a \in Act$ it holds that if $s_1 \xrightarrow{a} s_1'$ then there is a transition $s_2 \xrightarrow{a} s_2'$ such that $(s_1', s_2') \in R$, and if $s_2 \xrightarrow{a} s_2'$ then there is a transition $s_1 \xrightarrow{a} s_1'$ such that $(s_1', s_2') \in R$. Two states s and s' are *bisimilar*, written $s \sim s'$, iff there is a bisimulation R such that $(s, s') \in R$.

A finite *multiset* over some nonempty set A is a collection of elements from A where each element occurs in the multiset a finite amount of times; a multiset S over a set A can be identified with a function $S : A \rightarrow \mathbb{N}^0$ where $S(a)$ is the number of occurrences of element $a \in A$ in the multiset S. We shall represent multisets by a formal sum $\sum_{a \in A} S(a)'(a)$ such that e.g. $1'(x) + 2'(y)$ stands for a multiset containing one element x and two elements y. We assume the standard multiset operations of membership ($\in$), inclusion ($\subseteq$), equality ($=$), union ($\uplus$), subtraction ($\backslash$) and by $|S|$ we denote the cardinality of S (including the repetition of elements). By $\mathcal{S}(A)$ we denote the set of all multisets over the set A.

Finally, we also define the function *set* as a way of reducing multisets of colors to sets of colors given by $set(S) \stackrel{\text{def}}{=} \{a \mid a \in S\}$ where $set(S)$ is the set of all colors with at least one occurrence in S.

2.1. Colored Petri nets

Colored Petri nets (CPN) are an extension of traditional P/T nets introduced by Kurt Jensen [3] in 1981. In CPNs, places are associated with color domains where colors represent the values of tokens. Arc expressions describe what colors to consume and add to places depending on a given binding (assignment of variables to colors). Transitions may contain guards restricting which bindings are valid. There exist several different definitions of CPNs from the powerful version defined in [25] that includes the ML language for describing arcs expressions and guards to less powerful ones such as the one used in the Model Checking Contest [11]. We shall first give an abstract definition of a CPN.

Definition 2.1. A colored Petri net is a tuple $\mathcal{N} = (P, T, \mathbb{C}, \mathbb{B}, C, G, W, W_I, M_0)$ where

1. P is a finite set of places,

2. T is a finite set of transitions such that $P \cap T = \emptyset$,

3. $\mathbb{C}$ is a nonempty set of colors,

4. $\mathbb{B}$ is a nonempty set of bindings,

5. $C : P \to 2^{\mathbb{C}} \setminus \emptyset$ is a place color type function,

6. $G : T \times \mathbb{B} \to \{true, false\}$ is a guard evaluation function,

7. $W : ((P \times T) \cup (T \times P)) \times \mathbb{B} \to \mathcal{S}(\mathbb{C})$ is an arc evaluation function such that $set(W((p, t), b)) \subseteq C(p)$ and $set(W((t, p), b)) \subseteq C(p)$ for all $p \in P$, $t \in T$ and $b \in \mathbb{B}$,

8. $W_I : P \times T \to \mathbb{N}^{>0} \cup \{\infty\}$ is an inhibitor arc weight function, and

9. M_0 is the initial marking where a marking M is a function $M : P \to \mathcal{S}(\mathbb{C})$ such that $set(M(p)) \subseteq C(p)$ for all $p \in P$.

In this definition, we assume an abstract set of bindings $\mathbb{B}$, representing the concrete configurations a net can be in. For example, colored Petri nets often use variables on the arcs and in the guards, and these variables can be assigned concrete colors. Such a variable assignment is then referred to as a binding. Notice that G and W are semantic functions which are in different variants of CPN defined by a concrete syntax. These functions take as an argument a binding and for this binding return either whether the guard is true or false, or in case of W the function returns the multiset of tokens that should be consumed/produced when a transition is fired.

The set of all markings on a CPN $\mathcal{N}$ is denoted by $\mathbb{M}(\mathcal{N})$. In order to avoid the use of partial functions, we allow $W((p, t), b) = W((t, p), b) = \emptyset$ and $W_I(p, t) = \infty$, meaning that if the arc evaluation function returns the empty multiset then the arc has no effect on transition firing and if the inhibitor arc function returns infinity then it never inhibits the connected transition.

Let $\mathcal{N} = (P, T, \mathbb{C}, \mathbb{B}, C, G, W, W_I, M_0)$ be a fixed CPN for the rest of this section. Let $B(t) \overset{\text{def}}{=} \{b \in \mathbb{B} \mid G(t, b) = true\}$ be the set of all bindings that satisfy the guard of transition $t \in T$. Let $\ell : T \to Act$ be a transition labeling function. The semantics of a CPN $\mathcal{N}$ is defined as an LTS

$L(\mathcal{N}) = (\mathbb{M}(\mathcal{N}), Act, \rightarrow)$ where $\mathbb{M}(\mathcal{N})$ is the set of states defined as all markings on $\mathcal{N}$, Act is the set of actions, and $M \xrightarrow{a} M'$ iff there exists $t \in T$ where $\ell(t) = a$ and there is $b \in B(t)$ such that

$$W((p, t), b) \subseteq M(p) \text{ and } W_I(p, t) > |M(p)| \text{ for all } p \in P, \text{ and}$$
$$M'(p) = (M(p) \setminus W((p, t), b)) \uplus W((t, p), b) \text{ for all } p \in P.$$

We denote the firing of a transition $t \in T$ in marking M reaching M' as $M \xrightarrow{t} M'$. Let $\rightarrow = \bigcup_{t \in T} \xrightarrow{t}$ and let $\rightarrow^*$ be the reflexive and transitive closure of $\rightarrow$.

Remark 2.2. To reason about model checking of CPNs, we need to have a finite representation of colored nets that can be passed as an input to an algorithm. One way to enforce such a representation is to assume that all color domains are finite and the semantic functions C, G, W and W_I are effectively computable.

Finally, let us define the notion of postset and preset of $p \in P$ as $p^\bullet = \{t \in T \mid \exists b \in \mathbb{B}. \; W((p, t), b) \neq \emptyset\}$ and $^\bullet p = \{t \in T \mid \exists b \in \mathbb{B}. \; W((t, p), b) \neq \emptyset\}$. Similarly, for a transition $t \in T$ we define $t^\bullet = \{p \in P \mid \exists b \in \mathbb{B}. \; W((t, p), b) \neq \emptyset\}$ and $^\bullet t = \{p \in P \mid \exists b \in \mathbb{B}. \; W((p, t), b) \neq \emptyset\}$. We also define the preset of inhibitor arcs as $^\circ t = \{p \in P \mid W_I(p, t) \neq \infty\}$.

2.2. P/T nets

A Place/Transition (P/T) net is a CPN $\mathcal{N} = (P, T, \mathbb{C}, \mathbb{B}, C, G, W, W_I, M_0)$ with one color $\mathbb{C} = \{\bullet\}$ and only one binding $\mathbb{B} = \{b_\epsilon\}$ such that every guard evaluates to true i.e. $G(t, b_\epsilon) = \textit{true}$ for all $t \in T$ and every arc evaluates to a multiset over $\{\bullet\}$ i.e. $W((p, t), b_\epsilon) \in \mathcal{S}(\{\bullet\})$ and $W((t, p), b_\epsilon) \in \mathcal{S}(\{\bullet\})$ for all $p \in P$ and $t \in T$.

2.3. Integer colored Petri nets

An integer CPN (as used for example in the Model Checking Contest [11]) is a CPN

$$\mathcal{N} = (P, T, \mathbb{C}, \mathbb{B}, C, G, W, W_I, M_0)$$

where all colors are integer products i.e. $\mathbb{C} = \bigcup_{k \geq 1} (\mathbb{N}^0)^k$. We use interval ranges to describe sets of colors such that a tuple of ranges $([a_1, b_1], ..., [a_k, b_k])$ where $a_i, b_i \in \mathbb{N}^0$ for i, $1 \leq i \leq k$, describes the set of colors $\{(c_1, ..., c_k) \mid a_i \leq c_i \leq b_i \text{ for all } 1 \leq i \leq k\}$. If the interval upper-bound is smaller than the lower-bound, the interval range denotes the empty set and by $[a]$ we denote the singleton interval $[a, a]$. As an example, consider the place color type of some place p as $C(p) = [\![([1, 2], [6, 7])]\!]$ describing the set of colors $\{(1, 6), (1, 7), (2, 6), (2, 7)\}$. For notational convenience, we sometimes omit the semantic paranthesis and simply write $([1, 2], [6, 7])$ instead of $[\![([1, 2], [6, 7])]\!]$.

We use the set of variables $\mathcal{V} = \{x_1, ..., x_n\}$ to represent colors. Variables can be present on arcs and in guards. A binding $b : \mathcal{V} \rightarrow \mathbb{C}$ assigns colors to variables. We write $b \equiv \langle x_1 = c_1, ..., x_n = c_n \rangle$ for a binding where $b(x_i) = c_i$ for all i, $1 \leq i \leq n$. We now introduce the syntax of arc/guard expressions and its intuitive semantics by an example.

$$p_1 : (2'(0) + 3'(2)) + p_2 : 2'(5) \qquad \xrightarrow{t} \qquad p_1 : (1'(0) + 3'(2)) + p_2 : 1'(5) + p_3 : (1'((2,5)))$$
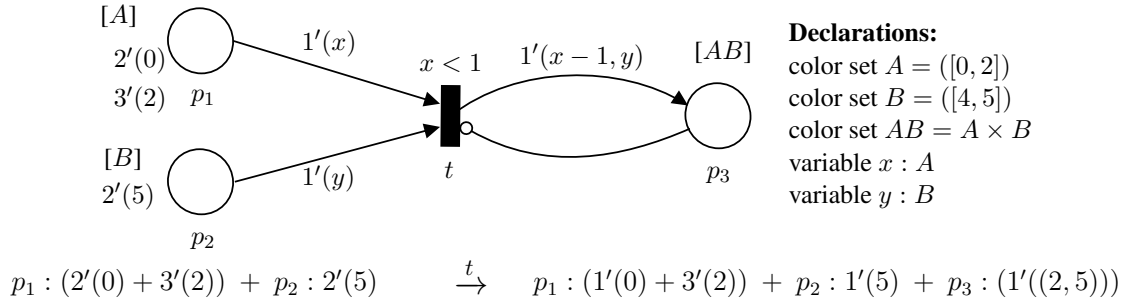
(a) Integer CPN and transition firing under the binding $\langle x = 0, y = 5 \rangle$
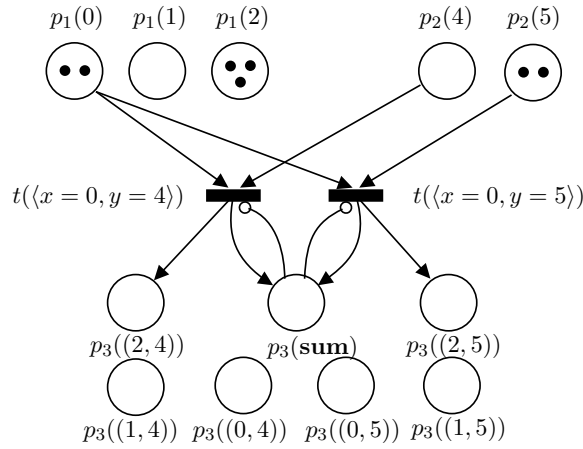
(b) Unfolding of CPN from Figure 1a

Figure 1: A CPN and its Unfolding to a P/T net

Figure 1a shows an integer CPN where places (circles) are associated with ranges. The initial marking contains five tokens (two of color 0 and three of color 2) in p_1 and two tokens of color 5 in place p_2. There is a guard on transition t (rectangle) that compares x with the integer 1 and restricts the valid bindings. We can see that the arc from t to p_3 creates a product of the integers x and y, where the value of x is decremented by one. We assume that all ranges are cyclic, meaning that the predecessor of 0 in the color set A is 2. Figure 1a also shows an example of transition firing. Markings are written as formal sums showing how many tokens of what colors are in the different places. The transition t can fire only once, as the inhibitor arc (for unlabelled inhibitor arcs we assume the default weight 1) from place p_3 to transition t inhibits the second transition firing.

We shall now present the formal syntax of arc expressions and guards. In integer CPNs, each arc $(P \times T) \cup (T \times P)$ excluding inhibitor arcs is assigned an arc expression α given by the syntax:

$$\alpha ::= n'(\tau_1, ..., \tau_k) \mid \alpha_1 \pm \alpha_2 \mid n \cdot \alpha$$
$$\tau ::= c \mid x \mid x \pm s$$

where $c \in \mathbb{C}$, $x \in \mathcal{V}$, $s \in \mathbb{N}^{>0}$, $n \in \mathbb{N}^{>0}$ and $\pm ::= + \mid -$. The α expressions allow us to make tuples consisting of τ expressions that can be combined using multiset union $(+)$ and subtraction $(-)$, or multiplied by n (creating n copies of the tuple). The expressions of type τ are called simple arc expressions. The semantics of arc expressions is straightforward and demonstrated by the following example.

Example 2.3. Let a be an arc annotated by the arc expression $1'(x-1) + 1'(y+1) + 1'(z)$ and let $b_1 = \langle x = 3, y = 3, z = 1 \rangle$ and $b_2 = \langle x = 1, y = 2, z = 2 \rangle$ be bindings with integer range $([1,3])$ over the variables x, y and z. The CPN semantics of the arc a is defined as the multiset where $W(a, b_1) = 1'(2) + 2'(1)$ since the colors are cyclic in nature such that $3 + 1 = 1$ and $W(a, b_2) = 2'(3) + 1'(2)$ because $1 - 1 = 3$.

Guards in integer CPNs are expressed by the following syntax:

$$\gamma ::= true \mid false \mid \neg\gamma \mid \gamma_1 \wedge \gamma_2 \mid \gamma_1 \vee \gamma_2 \mid \alpha_1 = \alpha_2 \mid \alpha_1 \neq \alpha_2 \mid \tau_1 \bowtie \tau_2$$

where $\bowtie ::=< \mid \leq \mid > \mid \geq \mid = \mid \neq$ and where α_1 and α_2 are arc expressions and τ_1 and τ_2 are simple arc expressions. The semantics of guards (their evaluation to true or false in a given binding) is also straightforward and demonstrated by an example.

Example 2.4. Let $g = (x > 2 \wedge y = 2) \vee z + 2 = 3$ be a guard on a transition t and let $b_1 = \langle x = 3, y = 3, z = 1 \rangle$ and $b_2 = \langle x = 1, y = 2, z = 2 \rangle$ be bindings with range $([1,3])$ over the variables x, y and z. Then $G(t, b_1) = true$ and $G(t, b_2) = false$.

The Model Checking Contest [11] further includes color types called dots and cyclic enumerations which are excluded from these definitions as these can be trivially translated to tuples of integer ranges. The color type dot, $\{\bullet\}$, is represented by the color domain $([1])$ and a cyclic enumeration with elements $\{e_1, e_2, ..., e_n\}$ is encoded as the integer range $([1, n])$ corresponding to the indices of the cyclic enumeration. Furthermore, the contest uses the *.all* expression, which creates one of each color in the color domain. For example if $A = ([0, 2])$ then $A.all = 1'(1) + 1'(2) + 1'(3)$.

All examples in this paper are expressed in integer CPN syntax.

2.4. Unfolding of CPNs

CPNs with finite color domains can be *unfolded* into an equivalent P/T net [26]. Each place p is unfolded into $|C(p)|$ places, a transition is made for each legal binding and we translate the multiset of colors on the arc to a multiset over $\bullet$. We now provide a formal definition of unfolding in our syntax, following the approach from [23, 27] that also consider inhibitor arcs.

For each place connected to an inhibitor arc, we create a fresh summation place that contains the sum of tokens across the rest of the unfolded places. The summation places are created to ensure that inhibitor arcs work correctly after unfolding.

Definition 2.5. (Unfolding)
Let $\mathcal{N} = (P, T, \mathbb{C}, \mathbb{B}, C, G, W, W_I, M_0)$ be a colored Petri net. The unfolded P/T net $\mathcal{N}^u = (P^u, T^u, \mathbb{C}^u, \mathbb{B}^u, C^u, G^u, W^u, W_I^u, M_0^u)$ is given by

1. $P^u = \{p(c) \mid p \in P \land c \in C(p)\} \cup \{p(\textbf{sum}) \mid t \in T, p \in {}^\circ t\}$,

2. $T^u = \bigcup_{t \in T} \bigcup_{b \in B(t)} t(b)$,

3. $\mathbb{C}^u = \{\bullet\}$,

4. $\mathbb{B}^u = \{b_\epsilon\}$,

5. $C^u(p(c)) = \{\bullet\}$ for all $p(c) \in P^u$,

6. $G^u(t(b), b_\epsilon) = \textit{true}$ for all $t(b) \in T^u$,

7. $W^u((p(c), t(b)), b_\epsilon) = W((p, t), b)(c)'(\bullet)$ and $W^u((t(b), p(c)), b_\epsilon) = W((t, p), b)(c)'(\bullet)$ for all $p(c) \in P^u$ and $t(b) \in T^u$, *and*
   $W^u((p(\textbf{sum}), t(b)), b_\epsilon) = |W((p, t), b)|'(\bullet)$ and $W^u((t(b), p(\textbf{sum})), b_\epsilon) = |W((t, p), b)|'(\bullet)$ for all $p(\textbf{sum}) \in P^u$ and $t(b) \in T^u$,

8. $W_I^u(p(\textbf{sum}), t(b)) = W_I(p, t)$ for all $p(\textbf{sum}) \in P^u$ and $t(b) \in T^u$, and

9. $M_0^u(p(c)) = M_0(p)(c)'(\bullet)$ for all $p(c) \in P^u$ *and*
   $M_0^u(p(\textbf{sum})) = |M_0(p)|'(\bullet)$ for all $p(\textbf{sum}) \in P^u$

where $p(\textbf{sum})$ denotes the sum of all tokens regardless of the color for the place p.

Consider the CPN in Figure 1a. The unfolded version of this can be seen in Figure 1b. We notice that each place of the CPN is unfolded to a fresh new place for every color in the color type of the place as well as a **sum** place for p_3. Additionally, the transition is unfolded to a new transition for each legal binding.

The theorem showing that the unfolded net is bisimilar to the original CPN was proved in [23, 27]; we only added a small optimization on the summation places.

Theorem 2.6. ([23, 27])
Given a colored Petri net $\mathcal{N} = (P, T, \mathbb{C}, \mathbb{B}, C, G, W, W_I, M_0)$ and the unfolded P/T net $\mathcal{N}^u = (P^u, T^u, \mathbb{C}^u, \mathbb{B}^u, C^u, G^u, W^u, W_I^u, M_0^u)$, it holds that $M_0 \sim M_0^u$ with the labeling function $\ell(t(b)) = t$ for all $t(b) \in T^u$.
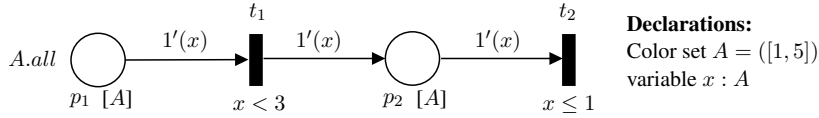
3. Color quotienting

Unfolding a CPN without any further analysis will often lead to many unnecessary places and transitions. We shall now present our first technique that allows to group equivalently behaving colors into equivalence classes in order to reduce the number of colors and hence also to reduce the size of the unfolded net.

As an example consider the CPN in Figure 2a, the unfolded version of this net adds five places for both p_1 and p_2. However, we see that in p_1 all colors greater than or equal to 3 behave exactly the same throughout the net and can thus be represented by a single color. We can thus *quotient* the CPN by *partitioning* the color domain of each place into a number of *equivalence classes* of colors such that the colors behaving equivalently are represented by the same equivalence class. Using this
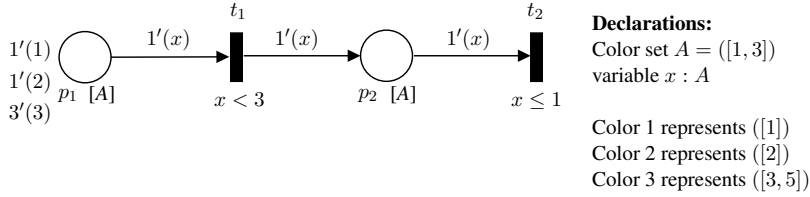
approach, we can construct a bisimilar CPN seen in Figure 2b where the color 3 now represents all colors greater than or equal to 3.

Such a reduction in the number of colors is possible to include already during the design of a CPN model, however, the models may look less intuitive for human modeller or the nets can be auto-generated and hence contain redundant/equivalent colors as observed in the benchmark of CPN models from the annual Model Checking Contest benchmark [11].

Declarations:
Color set $A = ([1, 5])$
variable $x : A$

(a) Example CPN

Declarations:
Color set $A = ([1, 3])$
variable $x : A$

Color 1 represents $([1])$
Color 2 represents $([2])$
Color 3 represents $([3, 5])$

(b) Quotiented net from Figure 2a

$$\delta(p_1) = \{([1, 2]), ([3, 5])\}, \delta(p_2) = \{([1]), ([2, 5])\}$$

$$M_1 = p_1 : 1'(1) \quad \xrightarrow{t_1} \quad M_1' = p_2 : 1'(1)$$

$$M_1 \stackrel{\delta}{\equiv} M_2 \qquad\qquad\qquad M_1' \stackrel{\delta}{\not\equiv} M_2'$$

$$M_2 = p_1 : 1'(2) \quad \xrightarrow{t_1} \quad M_2' = p_2 : 1'(2)$$

(c) Example of an unstable partition δ and markings showing why it is unstable

$$\delta'(p_1) = \{([1]), ([2]), ([3, 5])\}, \delta'(p_2) = \{([1]), ([2, 5])\}$$

(d) Example of stable partition δ'

Figure 2: Quotienting example

We thus introduce *color partition* on places where all colors with similar behaviour in a given place are grouped into an *equivalence class*, denoted by θ. For the rest of this section, let us assume a fixed CPN $\mathcal{N} = (P, T, \mathbb{C}, \mathbb{B}, C, G, W, W_I, M_0)$. A partition δ is a function $\delta : P \to 2^{2^{\mathbb{C}}} \setminus \emptyset$ that for a place p returns the equivalence classes of $C(p)$ such that $(\bigcup_{\theta \in \delta(p)} \theta) = C(p)$ and $\theta_1 \cap \theta_2 = \emptyset$ for all $\theta_1, \theta_2 \in \delta(p)$ where $\theta_1 \neq \theta_2$.

Definition 3.1. Given a partition δ and markings M and M', we write $M(p) \stackrel{\delta}{\equiv} M'(p)$ for a $p \in P$ iff for all $\theta \in \delta(p)$ it holds that $\sum_{c \in \theta} M(p)(c) = \sum_{c \in \theta} M'(p)(c)$. We write $M \stackrel{\delta}{\equiv} M'$ iff $M(p) \stackrel{\delta}{\equiv} M'(p)$ for all $p \in P$. A partition δ is *stable* if the relation $\stackrel{\delta}{\equiv}$ on markings induced by δ is a bisimulation.

Consider the CPN in Figure 2a. The partition shown in the Figure 2c is not stable as demonstrated by the transition firing from M_1 and M_2 to M_1' resp. M_2' where $M_1 \stackrel{\delta}{\equiv} M_2$ but $M_1' \stackrel{\delta}{\not\equiv} M_2'$. Figure 2d shows an example of a stable partition (here we describe the partition with integer ranges in the same manner as in integer CPNs).

We now show how a CPN can be quotiented using a stable partition. First, we define the notion of binding equivalence under a partition.

Definition 3.2. Given a partition δ, a transition $t \in T$ and bindings $b, b' \in B(t)$, we write $b \stackrel{\delta,t}{\equiv} b'$ iff for all $p \in {}^\bullet t$ and for all $\theta \in \delta(p)$ it holds that

$$\sum_{c \in \theta} W((p,t),b)(c) = \sum_{c \in \theta} W((p,t),b')(c)$$

and for all $p \in t^\bullet$ and for all $\theta \in \delta(p)$ it holds that

$$\sum_{c \in \theta} W((t,p),b)(c) = \sum_{c \in \theta} W((t,p),b')(c).$$

We can now define classes of equivalent bindings given a partition δ which are bindings that have the same behaviour for a given transition, formally $B^\delta(t) \stackrel{\text{def}}{=} \{[b]_t \mid b \in B(t)\}$ where $[b]_t = \{b' \mid b' \stackrel{\delta,t}{\equiv} b\}$.

For a given stable partition, we now construct a quotiented CPN where the set of colors are the equivalence classes of the stable partition and the set of bindings are the equivalence classes of bindings. As such, we rewrite the arc and guard evaluation functions to instead consider an equivalence class of bindings, which is possible since each binding in the equivalence class behaves equivalently.

Definition 3.3. Let $\mathcal{N} = (P, T, \mathbb{C}, \mathbb{B}, C, G, W, W_I, M_0)$ be a CPN and δ a stable partition of $\mathcal{N}$. The quotiented CPN $\mathcal{N}^\delta = (P, T, \mathbb{C}^\delta, \mathbb{B}^\delta, C^\delta, G^\delta, W^\delta, W_I^\delta, M_0^\delta)$ is defined by

1. $\mathbb{C}^\delta = \bigcup_{p \in P} \delta(p)$

2. $\mathbb{B}^\delta = \biguplus_{t \in T} B^\delta(t)$.

3. $G^\delta(t, [b]_t) = G(t, b)$ for all $t \in T$ and $[b]_t \in B(t)$,

4. $C^\delta(p) = \delta(p)$ for all $p \in P$,

5. $W^\delta((p,t), [b]_t) = S$ where $S(\theta) = \sum_{c \in \theta} W((p,t),b)(c)$ for all $\theta \in \delta(p)$ and
   $W^\delta((t,p), [b]_t) = S$ where $S(\theta) = \sum_{c \in \theta} W((t,p),b)(c)$ for all $\theta \in \delta(p)$
   for all $p \in P, t \in T$ and $[b]_t \in \mathbb{B}^\delta$,

6. $W_I^\delta(p, t) = W_I(p, t)$ for all $p \in P$ and $t \in T$, and

7. $M_0^\delta(p) = S$ where $S(\theta) = \sum_{c \in \theta} M_0(p)(c)$ for all $p \in P$ and $\theta \in \delta(p)$.

We can now present our main correctness theorem, stating that the original and quotiented colored nets are bisimilar.

Theorem 3.4. Let $\mathcal{N} = (P, T, \mathbb{C}, \mathbb{B}, C, G, W, W_I, M_0)$ be a CPN, δ a stable partition and $\mathcal{N}^\delta = (P^\delta, T^\delta, \mathbb{C}^\delta, \mathbb{B}^\delta, C^\delta, G^\delta, W^\delta, W_I^\delta, M_0^\delta)$ the quotiented CPN. Then $M_0 \sim M_0^\delta$.

Proof:
We show that $R = \{(M, M^\delta) \mid \sum_{c \in \theta} M(p)(c) = M^\delta(p)(\theta)$ for all $p \in P$ and all $\theta \in \delta(p)\}$ is a bisimulation relation. We first notice that $(M_0, M_0^\delta) \in R$ by Item 7 in Definition 3.3.

Assume that $(M, M^\delta) \in R$ and $t \in T$ such that $M \xrightarrow{t} M'$ under binding $b \in B(t)$, we want to show that $M^\delta \xrightarrow{t} M^{\delta\prime}$ under binding $[b]_t \in B(t)$ such that $(M', M^{\delta\prime}) \in R$. As such, we need to prove the following:

  (**a**) $W^\delta((p, t), [b]_t) \subseteq M^\delta(p)$ for all $p \in P$

  (**b**) $W_I^\delta(p, t) > |M^\delta(p)|$ for all $p \in P$

  (**c**) $(M', M^{\delta\prime}) \in R$ where $M^{\delta\prime}(p) = (M^\delta(p) \setminus W^\delta((p, t), [b]_t)) \uplus W^\delta((t, p), [b]_t)$
    for all $p \in P$.

(**a**) We start by showing $W^\delta((p, t), [b]_t) \subseteq M^\delta(p)$ for all $p \in P$. First, because $(M, M^\delta) \in R$, we know that

$$\sum_{c \in \theta} M(p)(c) = M^\delta(p)(\theta) \text{ for all } p \in P \text{ and all } \theta \in \delta(p). \tag{1}$$

Since $W((p, t), b) \subseteq M(p)$ we know for all $c \in C(p)$ that $W((p, t), b))(c) \le M(p)(c)$ which implies that

$$\sum_{c \in \theta} W((p, t), b))(c) \le \sum_{c \in \theta} M(p)(c) \tag{2}$$

for all $\theta \in \delta(p)$. We now show that $W^\delta((p, t), [b]_t) \subseteq M^\delta(p)$ for all $p \in P$ i.e. $W^\delta((p, t), [b]_t)(\theta) \le M^\delta(p)(\theta)$ for all $\theta \in \delta(p)$:

$W^\delta((p, t), [b]_t)(\theta)$	$=$	substitute by Def. 3.3 Item 5
$\sum_{c \in \theta} W((p, t), b))(c)$	$\le$	by Equation (2)
$\sum_{c \in \theta} M(p)(c)$	$=$	by Equation (1)
$M^\delta(p)(\theta)$.		

(**b**) Next we show $W_I^\delta((p, t) > |M^\delta(p)|$. We know that

$$W_I(p, t) > |M(p)| \tag{3}$$

by definition of CPN semantics since $M \xrightarrow{t} M'$. We then observe that for all $p \in P$, it holds

$W_I^\delta(p, t)$	$=$	substitute by Def. 3.3 Item 6
$W_I(p, t)$	$>$	by Equation (3)
$|M(p)|$	$=$	multiset definition
$\sum_{c \in C(p)} M(p)(c)$	$=$	since $(\bigcup_{\theta \in \delta(p)} \theta) = C(p)$
$\sum_{\theta \in \delta(p)} \sum_{c \in \theta} M(p)(c)$	$=$	by Equation (1)
$\sum_{\theta \in \delta(p)} M^\delta(p)(\theta)$	$=$	multiset definition
$|M^\delta(p)|$.		

(c) Lastly, we show that $(M', M^{\delta'}) \in R$. Assume $p \in P$, $b \in B(t)$ and equivalence class $[b]_t$. We know that $M'(p) = (M(p) \setminus W((p,t),b)) \uplus W((t,p),b)$ and $M^{\delta'}(p) = (M^{\delta}(p) \setminus W^{\delta}((p,t),[b]_t)) \uplus W^{\delta}((t,p),[b]_t)$ and we need to show that $\sum_{c \in \theta} M'(p)(c) = M^{\delta'}(p)(\theta)$ for all $\theta \in \delta(p)$:

$$\sum_{c \in \theta} M'(p)(c) \qquad = \qquad \text{by def. of CPN semantics}$$

$$\sum_{c \in \theta}(M(p) \setminus W((p,t),b) \uplus W((t,p),b))(c) \quad = \qquad \begin{array}{c}\text{substitute by multiset definitions}\\\text{and by enabledness of } t\end{array}$$

$$\begin{array}{c}\sum_{c \in \theta} M(p)(c) - \sum_{c \in \theta} W((p,t),b)(c)\\+\sum_{c \in \theta} W((t,p),b)(c)\end{array} \quad = \qquad \text{substitute by Def. 3.3 Item 5}$$

$$\begin{array}{c}\sum_{c \in \theta} M(p)(c) - W^{\delta}((p,t),[b]_t)(\theta)\\+W^{\delta}((t,p),[b]_t)(\theta)\end{array} \quad = \qquad \text{since } (M, M^{\delta}) \in R$$

$$\begin{array}{c}M^{\delta}(p)(\theta) - W^{\delta}((p,t),[b]_t)(\theta)\\+W^{\delta}((t,p),[b]_t)(\theta)\end{array} \quad = \qquad \text{by definition of CPN semantics}$$

$$M^{\delta'}(p)(\theta) \ .$$

We then have to show that the same is the case for the opposite direction such that assume $(M, M^{\delta}) \in R$ and $t \in T$ such that $M^{\delta} \xrightarrow{t} M^{\delta'}$, we want to show that $M \xrightarrow{t} M'$ for some $b \in B(t)$ such that $(M', M^{\delta'}) \in R$. As such, we want to show that:

(d) $W((p,t),b) \subseteq M(p)$ for all $p \in P$

(e) $W_I(p,t) > |M(p)|$ for all $p \in P$

(f) $(M', M^{\delta'}) \in R$ where $M'(p) = (M(p) \setminus W((p,t),b)) \uplus W((t,p),b)$
     for all $p \in P$.

We first notice that (e) and (f) can be showed by the same argumentation as (b) and (c). For the case (d), we show that $W((p,t),b) \subseteq M(p)$ for all $p \in P$. From (a) we know that $W^{\delta}((p,t),[b]_t) \subseteq M^{\delta}(p)$ which implies $\sum_{c \in \theta} W((p,t),b))(c) \leq \sum_{c \in \theta} M(p)(c)$ for all $\theta \in \delta(p)$ and $p \in P$.

Hence observe that there exists a marking M_1 such that $\sum_{c \in \theta} M_1(p)(c) = M^{\delta}(p)(\theta)$ and where $\sum_{c \in \theta} W((p,t),b)(c) \leq \sum_{c \in \theta} M_1(p)(c)$ for all $\theta \in \delta(p)$ and $p \in P$. Clearly t is enabled in M_1 since $W((p,t),b) \subseteq M_1(p)$ for all $p \in P$ by the multiset definition of $\subseteq$ and we know that the inhibitor arcs do not inhibit the transition by (e).

We then want to show that $M_1 \overset{\delta}{\equiv} M$, i.e. $\sum_{c \in \theta} M_1(p)(c) = \sum_{c \in \theta} M(p)(c)$ for all $\theta \in \delta(p)$ and $p \in P$. Since $(M, M^{\delta}) \in R$ we know that $\sum_{c \in \theta} M(p)(c) = M^{\delta}(p)(\theta) = \sum_{c \in \theta} M_1(p)(c)$ for all $\theta \in \delta(p)$ and $p \in P$ and thus $M_1 \overset{\delta}{\equiv} M$. Since δ is stable we know that t is enabled in M.

Thus we know that the opposite direction also holds meaning that R is a bisimulation. $\qquad\square$

3.1. Computing stable partitions

Our main challenge is how to efficiently compute a stable partition in order to apply the quotienting technique. To do so, we first define a partition refinement.

Definition 3.5. Given two partitions δ and δ' we write $\delta \geq \delta'$ if for all $p \in P$ and all $\theta' \in \delta'(p)$ there exists $\theta \in \delta(p)$ such that $\theta' \subseteq \theta$. Additionally, we write $\delta > \delta'$ if $\delta \geq \delta'$ and $\delta' \neq \delta$.

Note that for any finite CPN as assumed in Remark 2.2, the refinement relation $>$ is well-founded as for any $\delta > \delta'$ the partition δ' has strictly more equivalence classes for at least one place $p \in P$. We now define also the union of two partitions as the smallest partition that has both of the partitions as refinements.

Definition 3.6. Given two partitions δ_1, δ_2 and $p \in P$, let $\leftrightarrow$ be a relation over $\delta_1(p) \cup \delta_2(p)$ such that $\theta \leftrightarrow \theta'$ iff $\theta \cap \theta' \neq \emptyset$ where $\theta, \theta' \in \delta_1(p) \cup \delta_2(p)$. Let $\leftrightarrow^*$ be the reflexive, transitive closure of $\leftrightarrow$ and let $[\theta] \overset{\text{def}}{=} \bigcup_{\theta' \in \delta_1(p) \cup \delta_2(p), \theta \leftrightarrow^* \theta'} \theta'$ where $\theta \in \delta_1(p) \cup \delta_2(p)$. Finally, we define the partition union operator $\sqcup$ by $(\delta_1 \sqcup \delta_2)(p) = \bigcup_{\theta \in \delta_1(p) \cup \delta_2(p)} \{[\theta]\}$ for all $p \in P$.

For example, assume some place p such that $C(p) = \{([1,5])\}$ and partitions δ_1 and δ_2 such that $\delta_1(p) = \{([1,2]), ([3,4]), ([5])\}$ and $\delta_2(p) = \{([1]), ([2,3]), ([4]), ([5])\}$ then $(\delta_1 \sqcup \delta_2)(p) = \{([1,4]), ([5])\}$.

Lemma 3.7. Let δ_1 and δ_2 be two partitions. Then (i) $\delta_1 \sqcup \delta_2 \geq \delta_1$ and $\delta_1 \sqcup \delta_2 \geq \delta_2$, and (ii) if δ_1 and δ_2 are stable partitions then so is $\delta_1 \sqcup \delta_2$.

Proof:
For the first part of the claim, from definition of partition union, we see that $[\theta]$ is the union of any θ' that overlaps with θ and $\delta_1 \sqcup \delta_2$ just collects all such unions for every θ. As such, it is trivial that for any $\theta \in \delta_1(p)$ there exists $\theta' \in \delta_1(p) \sqcup \delta_2(p)$ such that $\theta \subseteq \theta'$ for all $p \in P$, in other words $\delta_1 \sqcup \delta_2 \geq \delta_1$. The same is the case for δ_2.

For the second part of the claim, let $\delta = \delta_1 \sqcup \delta_2$. Assume M and M' such that $M \overset{\delta}{\equiv} M'$ i.e. for all $p \in P$ and all $[\theta] \in \delta(p)$ it holds that $\sum_{c \in [\theta]} M(p)(c) = \sum_{c \in [\theta]} M'(p)(c)$. We want to show that δ is stable, i.e that $\overset{\delta}{\equiv}$ is a bisimulation relation, while assuming that $\overset{\delta_1}{\equiv}$ and $\overset{\delta_2}{\equiv}$ are bisimulation relations. Assume a fixed $p \in P$. From definition of partition union, we get that for every fixed $[\theta] \in \delta(p)$ and for all $c, c' \in [\theta]$ there exist $c_1, ..., c_k \in [\theta]$ such that $c, c_1 \in \theta_1$, $c_1, c_2 \in \theta_2$, $c_2, c_3 \in \theta_3$, $\ldots$, $c_k, c' \in \theta_k$, where $\theta_i \in \delta_1(p) \cup \delta_2(p)$ for all $i, 1 \leq i \leq k$. Let us define markings M_i, for $1 \leq i \leq k$, such that $M_i(p)(c_i) = \sum_{c \in [\theta]} M(p)(c)$ and $M_i(p)(c) = 0$ if $c \neq c_i$ and $c \in [\theta]$, otherwise $M_i(p)(c) = M(p)(c)$. In other words, in order to obtain M_i, we replace in M all colors in the equivalence class $[\theta]$ with the color c_i and obtain the chain $M(p) \overset{\delta_{j1}}{\equiv} M_1(p) \overset{\delta_{j2}}{\equiv} M_2(p) \overset{\delta_{j3}}{\equiv} \ldots \overset{\delta_{jk}}{\equiv} M_k(p)$ where $j_i \in \{1, 2\}$, i.e. M_i and M_{i+1} are related either by $\overset{\delta_1}{\equiv}$ or $\overset{\delta_2}{\equiv}$, both of them being bisimulation relations. We can also observe that $M(p) \overset{\delta}{\equiv} M_k(p)$. We can then repeat this process for all other equivalence classes in $\delta(p)$ in order to conclude that $M(p) \overset{\delta}{\equiv} M'(p)$. The same process can then be applied to all $p \in P$, implying that $M \overset{\delta}{\equiv} M'$. Hence there is a chain of markings as above starting from M and ending in M'. Since δ_1 and δ_2 are stable, meaning that both $\overset{\delta_1}{\equiv}$ and $\overset{\delta_2}{\equiv}$ are bisimulation relations, this implies that every transition from M can be (by transitivity) matched by a transition from M' and vice versa, implying that $\overset{\delta}{\equiv}$ is a bisimulation relation and δ is so stable.   □

The lemma above implies the existence of a unique maximum stable partition.

Theorem 3.8. There is a unique maximum stable partition δ such that $\delta \geq \delta'$ for all stable partitions δ'.

Proof:
We prove this by contradiction. Assume two maximum stable partitions δ_1 and δ_2 where $\delta_1 \neq \delta_2$. By Lemma 3.7 (ii) we know that $\delta_1 \sqcup \delta_2$ is stable and by Lemma 3.7 (i) we know that $\delta_1 \sqcup \delta_2 \geq \delta_1$ and $\delta_1 \sqcup \delta_2 \geq \delta_2$. Thus δ_1 and δ_2 cannot both be maximum stable partitions. □

In order to provide an algorithm for computing a stable partition, we define the maximum arc size for a given CPN $\mathcal{N}$ as the function $max(\mathcal{N}) = \max_{p \in P, t \in T, b \in \mathbb{B}}(|W((p,t),b)|, |W((t,p),b)|)$. The set of all markings smaller than the *max* arc size over $\mathcal{N}$ is defined by $\mathbb{M}^{bounded}(\mathcal{N}) = \{M \in \mathbb{M}(\mathcal{N}) \mid |M(p)| \leq max(\mathcal{N})$ for all $p \in P\}$. As such, $\mathbb{M}^{bounded}(\mathcal{N})$ is a finite set of all bounded markings of $\mathcal{N}$ with cardinality less than or equal to $max(\mathcal{N})$.

In order to compute stable partitions we need to show some properties for markings in $\mathbb{M}^{bounded}(\mathcal{N})$. The following lemma shows that if there are two non-bisimilar markings that break the fact that the relation $\overset{\delta}{\equiv}$ is a bisimulation, then there are also two markings that are bounded and that also break the bisimulation property.

Lemma 3.9. Let $\mathcal{N}$ be a CPN and δ a partition. Then for all $t \in T$ it holds that

$(\boldsymbol{a})$ if there exist $M_1, M_2 \in \mathbb{M}(\mathcal{N})$ such that $M_1 \overset{\delta}{\equiv} M_2$, $M_1 \overset{t}{\rightarrow}$ and $M_2 \overset{t}{\nrightarrow}$ then there exist $M_3, M_4 \in \mathbb{M}^{bounded}$ such that $M_3 \overset{\delta}{\equiv} M_4$, $M_3 \overset{t}{\rightarrow}$ and $M_4 \overset{t}{\nrightarrow}$, and

$(\boldsymbol{b})$ if there exist $M_1, M_2 \in \mathbb{M}(\mathcal{N})$ where $M_1 \overset{\delta}{\equiv} M_2$ and there exists $M_1' \in \mathbb{M}(\mathcal{N})$ such that $M_1 \overset{t}{\rightarrow} M_1'$ and for all $M_2' \in \mathbb{M}(\mathcal{N})$ where $M_2 \overset{t}{\rightarrow} M_2'$ it holds that $M_1' \overset{\delta}{\nequiv} M_2'$ then there exists $M_3, M_4 \in \mathbb{M}^{bounded}(\mathcal{N})$ where $M_3 \overset{\delta}{\equiv} M_4$ and there exists $M_3' \in \mathbb{M}^{bounded}(\mathcal{N})$ such that $M_3 \overset{t}{\rightarrow} M_3'$ and for all $M_4' \in \mathbb{M}^{bounded}(\mathcal{N})$ where $M_4 \overset{t}{\rightarrow} M_4'$ it holds that $M_3' \overset{\delta}{\nequiv} M_4'$.

Proof:
Recall that $max(\mathcal{N})$ is defined as the largest cardinality of all arc multisets in $\mathcal{N}$, i.e. $|W((p,t),b)| \leq max(\mathcal{N})$ for all $p \in P$ and $b \in B(t)$.

$(\boldsymbol{a})$ Let $M_1, M_2 \in \mathbb{M}(\mathcal{N})$ such that $M_1 \overset{\delta}{\equiv} M_2$, $M_1 \overset{t}{\rightarrow}$ and $M_2 \overset{t}{\nrightarrow}$. We construct a marking M_3 such that $M_3(p) = W((p,t),b)$ for all $p \in P$ and some $b \in B(t)$. It clearly follows that $|M_3(p)| \leq max(\mathcal{N})$ for all $p \in P$, hence $M_3 \in \mathbb{M}^{bounded}(\mathcal{N})$. We see that $M_1(p) = M_3(p) \uplus \overline{M}_3(p)$ since $M_3(p) \subseteq M_1(p)$ for all $p \in P$ where $\overline{M}_3(p)$ describes the remaining tokens in $M_1(p)$ that are not in $M_3(p)$. We know that no inhibitor arc can be the reason that M_2 is not enabled because $M_1 \overset{\delta}{\equiv} M_2$. We also know that $M_2(p) \not\subseteq W((p,t),b)$ for at least one $p \in P$ for all $b \in B(t)$.

We pick a marking M_4 where $M_4 \stackrel{\delta}{\equiv} M_3$, $M_4(p) \not\subseteq W((p,t),b)$ for at least one $p \in P$ and $M_2(p) = M_4(p) \uplus \overline{M}_4(p)$ for all $p \in P$ such that $\overline{M}_4 \stackrel{\delta}{\equiv} \overline{M}_3$. Notice that $M_4 \in \mathbb{M}^{bounded}(\mathcal{N})$. We know that M_4 exists because $M_2 \stackrel{\delta}{\equiv} M_1$ and $M_2(p) \not\subseteq W((p,t),b)$ meaning that $M_4(p) \uplus \overline{M}_4(p) \not\subseteq W((p,t),b)$ and thus $M_4(p) \not\subseteq W((p,t),b)$ for some $p \in P$.

(**b**) Let $M_1, M_2 \in \mathbb{M}(\mathcal{N})$ such that $M_1 \stackrel{\delta}{\equiv} M_2$ and $M_1' \in \mathbb{M}(\mathcal{N})$ such that $M_1 \stackrel{t}{\to} M_1'$ while for all $M_2' \in \mathbb{M}(\mathcal{N})$ where $M_2 \stackrel{t}{\to} M_2'$ we know that $M_1' \stackrel{\delta}{\not\equiv} M_2'$. We construct a marking M_3 exactly as before such that $M_3(p) = W((p,t),b)$ for all $p \in P$ and some $b \in B(t)$ and $M_1(p) = M_3(p) \uplus \overline{M}_3(p)$ for all $p \in P$.

We then pick a marking M_4 such that $M_4 \stackrel{\delta}{\equiv} M_3$ and $M_2(p) = M_4(p) \uplus \overline{M}_4(p)$ for all $p \in P$ and $b \in B(t)$ such that $\overline{M}_4 \stackrel{\delta}{\equiv} \overline{M}_3$. We know that $M_4 \in \mathbb{M}^{bounded}(\mathcal{N})$ since $M_4 \stackrel{\delta}{\equiv} M_3$. Let $M_3' \in \mathbb{M}^{bounded}(\mathcal{N})$ such that $M_3 \stackrel{t}{\to} M_3'$, which is possible because $M_3(p) \subseteq M_1(p)$ for all $p \in P$ such that no inhibitor arc can inhibit M_3. For the sake of contradiction now assume there exists a marking $M_4' \in \mathbb{M}^{bounded}(\mathcal{N})$ such that $M_4 \stackrel{t}{\to} M_4'$ and $M_3' \stackrel{\delta}{\equiv} M_4'$. Then notice that we can let $M_1'(p) = M_3'(p) \uplus \overline{M}_3(p)$ where $M_1 \stackrel{t}{\to} M_1'$ since $M_3(p) \subseteq M_1(p)$ for all $p \in P$ and let $M_2'(p) = M_4'(p) \uplus \overline{M}_4(p)$ where $M_2 \stackrel{t}{\to} M_2'$ since $M_4(p) \subseteq M_2(p)$ for all $p \in P$. But since $M_3'(p) \stackrel{\delta}{\equiv} M_4'(p)$ and $\overline{M}_3(p) \stackrel{\delta}{\equiv} \overline{M}_4(p)$ it means that $M_3'(p) \uplus \overline{M}_3(p) \stackrel{\delta}{\equiv} M_4'(p) \uplus \overline{M}_4(p)$ for all $p \in P$, i.e. $M_1' \stackrel{\delta}{\equiv} M_2'$. However, this contradicts the conditions of (**b**), and as such $M_3' \stackrel{\delta}{\equiv} M_4'$ cannot hold. $\square$

Algorithm 1 now gives a procedure for computing a stable partition over a given CPN. It starts with an initial partition where every color in the color domain is in the same equivalence class for each place. The algorithm is then split into two parts. The first part from line 4 to 9 creates an initial partition applying the guard restrictions to the input places of the transitions. The second part from line 11 to 20 back propagates the guard restrictions throughout the net such that only colors that behave the same are quotiented together. Depending on the choices in lines 6 and 15, the algorithm may return the maximum stable partition, however in the practical implementation this is not guaranteed due to an approximation of the guard/arc expression analysis.

Theorem 3.10. Given a CPN $\mathcal{N}$, the algorithm *Stabilize*($\mathcal{N}$) terminates and returns a stable partition of $\mathcal{N}$.

Proof:
We first prove that *Stabilize*($\mathcal{N}$) terminates. Notice that each iteration produces a new δ according to the $>$ operator, and since the operator is well-founded we know that the algorithm terminates. We then show that for $\delta = $ *Stabilize*($\mathcal{N}$), δ is a stable partition of $\mathcal{N}$. Recall, a partition δ is stable iff for any markings $M_1 \stackrel{\delta}{\equiv} M_2$ whenever $M_1 \stackrel{t}{\to} M_1'$ for some t and M_1' then $M_2 \stackrel{t}{\to} M_2'$ for some M_2' such that $M_1' \stackrel{\delta}{\equiv} M_2'$. We prove this by contradiction. Assume that δ is not a stable partition. As such there must exists markings $M_1, M_2 \in \mathbb{M}(\mathcal{N})$ such that $M_1 \stackrel{\delta}{\equiv} M_2$ and there exists a marking $M_1' \in \mathbb{M}(\mathcal{N})$

Algorithm 1: *Stabilize*($\mathcal{N}$)

1 **Input**: $\mathcal{N} = (P, T, \mathbb{C}, \mathbb{B}, C, G, W, W_I, M_0)$
2 **Output**: Stable partition δ
3 let $\delta(p) := \{C(p)\}$ for all $p \in P$
4 **for** $t \in T$ **do**
5   **while** $\exists M_1, M_2 \in \mathbb{M}^{bounded}(\mathcal{N}).M_1 \overset{\delta}{\equiv} M_2 \wedge M_1 \overset{t}{\not\rightarrow} \wedge M_2 \overset{t}{\rightarrow}$ **do**
6   pick $\delta' < \delta$ such that $M_1 \overset{\delta'}{\not\equiv} M_2$
7        $\delta := \delta'$
8   **end**
9 **end**
10 let $\mathcal{Q} := P$ //Waiting list of places
11 **while** $\mathcal{Q} \neq \emptyset$ **do**
12   let $p \in \mathcal{Q}$; $\mathcal{Q} := \mathcal{Q} \setminus \{p\}$
13   **for** $t \in {}^\bullet p$ **do**
14   **if** $\exists M_1, M_2 \in \mathbb{M}^{bounded}(\mathcal{N}).M_1 \overset{\delta}{\equiv} M_2.\exists M_1' \in \mathbb{M}^{bounded}(\mathcal{N}).M_1 \overset{t}{\rightarrow} M_1' \wedge \forall M_2' \in$
            $\mathbb{M}^{bounded}(\mathcal{N}).M_2 \overset{t}{\rightarrow} M_2' \wedge M_1'(p) \overset{\delta}{\not\equiv} M_2'(p)$ **then**
15   pick $\delta' < \delta$ such that $M_1 \overset{\delta'}{\not\equiv} M_2$ and $\delta'(p') = \delta(p')$ for all $p' \in P \setminus {}^\bullet t$
16           $\mathcal{Q} := \mathcal{Q} \cup \{p' \mid \delta'(p') \neq \delta(p')\}$
17           $\delta := \delta'$
18   **end**
19   **end**
20 **end**
21 **return** δ

---

such that $M_1 \overset{t}{\rightarrow} M_1'$ for some transition t where for all $M_2' \in \mathbb{M}(\mathcal{N})$ such that $M_2 \overset{t}{\rightarrow} M_2'$ then $M_1' \overset{\delta}{\not\equiv} M_2'$. This is exactly the property stated in the if statement on line 17 and we know from Lemma 3.9 that if the property is satisfied with two markings from $\mathbb{M}(\mathcal{N})$ then there exists two markings from $\mathbb{M}^{bounded}(\mathcal{N})$ that also satisfy the property. $\square$

3.2. Stable partition Algorithm for integer CPNs

The *Stabilize* computation presented in Algorithm 1 can be used to find a stable partition for any finite CPN. However, implementation-wise it is inefficient to represent every color in a given equivalence class individually. Hence, for integer CPN we represent an equivalence class as a tuple of ranges. As an example of computing stable partitions with Algorithm 1, consider the integer CPN in Figure 3. Table 1 shows the different stages that δ undergoes in order to become stable. In iteration 0, the guard restrictions from the first for-loop are applied, followed by the iterations of the main while-loop. In our implementation, we do not iterate through every bounded marking and we instead (for efficiency
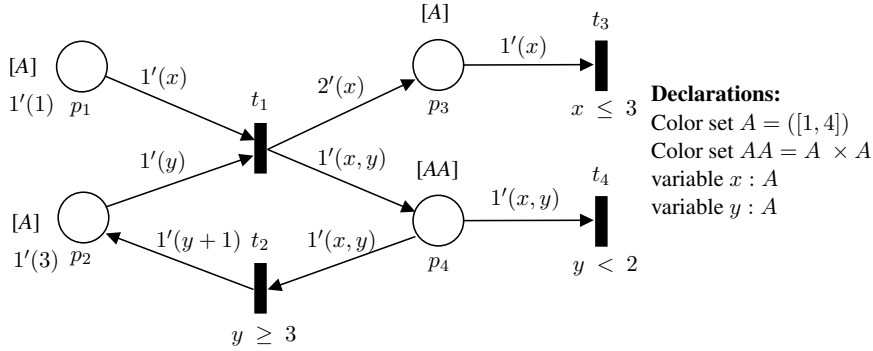
Figure 3: Example CPN

Table 1: Stages of δ throughout Algorithm 1 for CPN in Figure 3. The 0'th iteration is the state of δ just before the while loop begins. The symbol '-' indicates that the value is the same as in the previous row.

Iteration	p_1	p_2	p_3	p_4	Q
0	$\{([1,4])\}$	$\{([1,4])\}$	$\{([1,3]),([4])\}$	$\{([1,4],[1]),([1,4],[2]),([1,4],[3,4])\}$	$\{p_1,p_2,p_3,p_4\}$
$1, p = p_3$	$\{([1,3]),([4])\}$	-	-	-	$\{p_1,p_2,p_4\}$
$2, p = p_4$	-	$\{([1]),([2]),([3,4])\}$	-	-	$\{p_1,p_2\}$
$3, p = p_2$	-	-	-	$\{([1,4],[1]),([1,4],[2]),$ $([1,4],[3]),([1,4],[4])\}$	$\{p_1,p_4\}$
$4, p = p_4$	-	$\{([1]),([2]),([3]),([4])\}$	-	-	$\{p_1,p_2\}$
$5, p = p_2$	-	-	-	-	$\{p_1\}$
$6, p = p_1$	-	-	-	-	$\{\}$

Table 2: Stages of α when computing the fixed point of E for the CPN in Figure 3. The symbol '-' indicates that the value is the same as in the previous row.

Iteration	p_1	p_2	p_3	p_4
$0, \alpha = \alpha_0$	$\{([1])\}$	$\{([3])\}$	$\{\}$	$\{\}$
$1, t = t_1$	-	-	$\{([1])\}$	$\{([1],[3])\}$
$2, t = t_2$	-	$\{([3,4])\}$	-	-
$3, t = t_1$	-	-	-	$\{([1],[3,4])\}$
$4, t = t_2$	-	$\{([3,4]),([1])\}$	-	-
$5, t = t_1$	-	-	-	$\{([1],[3,4]),([1],[1])\}$

reasons) statically analyze the places, arcs and guards in order to partition the color sets. For example, in iteration number 1, we consider the place p_3 and we can see that the colors in the range $[1,3]$ must be distinguished from the color 4. This partitioning propagates back to the place p_1 as firing the transition t_1 moves tokens from p_1 to p_3 without changing its color.

4. Color approximation

We now introduce another technique for safely overapproximating what colors can be present in each place of a CPN. Let $\mathcal{N} = (P, T, \mathbb{C}, \mathbb{B}, C, G, W, W_I, M_0)$ be a fixed CPN for the rest of this section. A *color approximation* is a function $\alpha : P \to 2^{\mathbb{C}}$ where $\alpha(p)$ approximates the possible colors in place $p \in P$ such that $\alpha(p) \subseteq C(p)$. Let $\mathbb{A}$ be the set of all color approximations. For a marking M and color approximation α, we write $M \subseteq \alpha$ iff $set(M(p)) \subseteq \alpha(p)$ for all $p \in P$. A *color expansion* is a function $E : \mathbb{A} \to \mathbb{A}$ defined by

$$
E(\alpha)(p) = \begin{cases} \alpha(p) \cup set(W((t,p),b)) & \text{if } \exists t \in T. \exists b \in B(t). \\ & set(W((q,t),b)) \subseteq \alpha(q) \text{ for all } q \in P \\ \alpha(p) & \text{otherwise.} \end{cases}
$$

A color expansion iteratively expands the possible colors that exist in each place but without keeping a count of how many tokens of each color are present (if a color is present in a place, we assume that arbitrary many copies of the color are present). The expansion function obviously preserves the following property.

Lemma 4.1. Let α be a color approximation then $\alpha(p) \subseteq E(\alpha)(p)$ for all $p \in P$.

Let α_0 be the initial approximation such that $\alpha_0(p) \stackrel{\text{def}}{=} set(M_0(p))$ for all $p \in P$. Since E is a monotonic function on a complete lattice, we can compute its minimum fixed point and formulate the following key lemma.

Lemma 4.2. Let α be a minimum fixed point of E such that $\alpha_0(p) \subseteq \alpha(p)$ for all $p \in P$. If $M_0 \to^* M$ then $M \subseteq \alpha$.

Proof:
By induction on k we prove if $M_0 \to^k M$ then $M \subseteq \alpha$. *Base step.* If $k = 0$, we know that $M_0 \subseteq \alpha$ by the assumption of the lemma. *Induction step.* Let $M_0 \to^k M \xrightarrow{t} M'$ by some transition t with some binding $b \in B(t)$. We want to show that $M' \subseteq \alpha$. By induction hypothesis we know that $M \subseteq \alpha$. If $M \xrightarrow{t} M'$ for some $b \in B(t)$, then $M'(p) = (M(p) \setminus W((p,t),b)) \uplus W((t,p),b)$ for all $p \in P$. Since $E(\alpha)$ is a fixed point then $\alpha(p) = \alpha(p) \cup set(W((t,p),b))$ for transition t under binding b for all $p \in P$ i.e. $set(W((q,t),b)) \subseteq \alpha(q)$ for all $q \in P$. Thus we get $M' \subseteq \alpha$. □

Given a color approximation α satisfying the preconditions of Lemma 4.2, we can now construct a reduced CPN $\mathcal{N}^\alpha = (P, T, \mathbb{C}, \mathbb{B}, C^\alpha, G, W, W_I, M_0)$ where $C^\alpha(p) = \alpha(p)$ for all $p \in P$. The net $\mathcal{N}^\alpha$ can hence have possibly smaller set of colors in its color domains and it satisfies the following theorem.

Theorem 4.3. The reachable fragments from the initial marking M_0 of the LTSs generated by $\mathcal{N}$ and $\mathcal{N}^\alpha$ are isomorphic.

Proof:

By Lemma 4.2 we know that $M \subseteq \alpha$ for any marking $M \in \mathbb{M}(\mathcal{N})$ reachable from M_0. Since the reachable fragments of $\mathcal{N}$ and $\mathcal{N}^\alpha$ are exactly the reachable markings, we know that the reachable fragments are isomorphic. □

4.1. Computing color approximation on integer CPNs

As with color quotienting, representing each color individually becomes inefficient. We thus employ integer ranges to represent color approximations. Consider the approximation α where $\alpha(p) = \{(1, 2), (2, 2), (3, 2), (5, 6), (5, 7)\}$ are possible colors (pairs of integers) in the place p; this can be more compactly represented as a set of tuples of ranges $\{([1, 3], [2]), ([5], [6, 7])\}$.

However, computing the minimum fixed point of E using ranges is not as trivial as using complete color sets. To do so, we need to compute new ranges depending on arcs and guards. We demonstrate this on the CPN in Figure 3. Table 2 shows the computation of the minimum fixed point of E, starting from the initial approximation α_0. For example, in iteration number 5, we check if firing transition t_1 can produce any additional tokens to the places p_3 and p_4. Clearly, there is no change to the possible token colors in p_3 as $\alpha(p_1)$ did not change, however the addition of the integer range $[1]$ to $\alpha(p_2)$ in the previous iteration now allows us to produce a new token color $(1, 1)$ into p_4 and hence we add the singleton range $([1], [1])$ to $\alpha(p_4)$. Due to the guard $y \geq 3$ on the transition t_2, we know that the added token $(1, 1)$ does not generate any further behaviour and hence we reached a fixed point.

5. Experiments

We implemented the quotienting method from Section 3 as well as the color approximation method from Section 4 in C++ as an extension to the verification engine *verifypn* [13] from the TAPAAL toolchain [12].

We perform experiments by comparing several different approaches; the quotienting approach (method A), the color approximation approach (method B) and the combination of both (method A+B) against

- the unfolder MCC [4] (used also by TINA [17] and LoLA [18]),

- ITS-Tools unfolder [20],

- the Spike unfolder [16] (also used by MARCIE [15] and Snoopy [14]),

- and *verifypn* TAPAAL unfolder with methods A and B disabled, referred to as Tapaal.

We compare the tools on the complete set of CPN nets and queries from 2021 Model Checking Contest [11][1]. The experiments are conducted on a compute cluster running Linux version 5.4.0, where each experiment is conducted on a AMD Epyc 7642 processor with a 15 GB memory limit and 5 minute timeout. To reduce noise in the experiments, we read and write models to a RAM-disk (not included in the 15 GB). A repeatability package is available in [28].

---

[1]We omit the *UtilityControlRoom* family of models as they contain operators not supported by ITS-Tools.

Table 3: Number of unfolded nets for each unfolder

	Spike	Tapaal	ITS-Tools	A	B	MCC	A+B	Total
Unfolded nets	170	192	193	195	200	200	203	203

We conduct two series of experiments: in Section 5.1 we study the unfolding performance of the tools; i.e. how fast and how many models can be unfolded by each tool and how large are the unfolded models, and in Section 5.2 we study the impact of this unfolding on the ability to answer the queries from the Model Checking Contest 2021.

5.1.   Unfolding experiments

Table 3 shows for each of the unfolders the number of unfolded nets within the memory/time limit. The last column shows the total number of unfolded nets by all tools combined, and we notice that the combination of methods A+B allows us to unfold a superset of models unfolded by the other tools. Our method A+B can unfold 3 nets that no other tool can unfold; DrinkVendingMachine48, 76, 98. This is directly attributed to method A.

The comparison of the sizes (total number of transitions and places) of unfolded nets is done by plotting the ratios between the size produced by our A+B method and the competing unfolder. Figure 4a shows the size ratios where at least one comparison is not equal to 1. We see that our method has a size ratio always smaller or equal to 1 (no other method unfolds any of the nets to a smaller size compared to our method A+B) and the size ratio is strictly smaller than 1 for 144 colored nets (out of 213 nets in the database). Moreover, we can reduce 45 nets by at least one order of magnitude, compared to all other unfolders.

As our method outperforms the state-of-the-art unfolders w.r.t. the size of the unfolded nets, the question is whether the overhead of the advanced static analysis does not kill the benefits. Fortunately, this is not the case as shown in Figure 4b where the 80 slowest unfolding times (independently sorted in nondecreasing order for each tool) are depicted. The plots show that our method has almost a magnitude reduction in running-time compared to ITS-Tools and MCC (which are close in performance), while Spike is significantly slower. ITS-Tools is generally fast on the nets that are unfolded in less than 10 seconds, however it becomes gradually slower and has problems unfolding the larger nets. Our unfolder and the MCC unfolder demonstrate a similar degradation trend in performance.

The overall conclusion is that our advanced analyses requires less overhead compared to other existing unfolders, while at the same time it significantly decreases the size of the unfolded nets.

5.2.   Query verification experiments

In these experiments, we examine which unfolding engine allows for most query answers verified on the unfolded nets. To allow for a fair comparison, we let each tool unfold and output the net to a PNML file. Regarding queries, both method A+B and ITS-Tools can already output the unfolded queries. For MCC, we implement our own translation from the colored queries to the unfolded queries
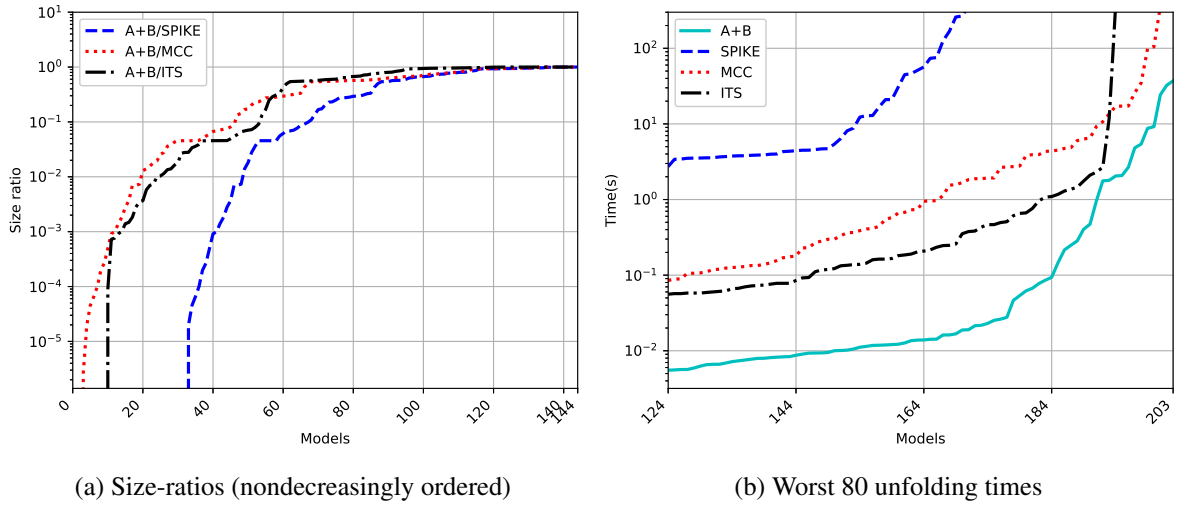
(a) Size-ratios (nondecreasingly ordered)   (b) Worst 80 unfolding times

Figure 4: Unfolding size and unfolding time comparison

for the given nets. For Spike, we were not able to construct a query unfolder that worked consistently and for this reason Spike is excluded from the query verification experiments.

Since we are testing the effect of the unfolding and not the verification engine, we use *verifypn* (revision 507c8ee0) to verify the queries on the nets unfolded by the different unfolders. There is a total of 20,448 queries to be answered. The results can be seen in Table 4.

Table 4: Number of queries answered for the unfolded nets of each tool. The % column describes how many percent of the total available queries in each category were answered. Each category counts $213 \cdot 16 = 3408$ queries.

Cardinality Queries						
	A+B		MCC		ITS-Tools	
	Solved	%	Solved	%	Solved	%
ReachabilityCardinality	2909	85.4	2793	82.0	2759	81.0
CTLCardinality	2651	77.8	2490	73.1	2443	73.8
LTLCardinality	2952	86.6	2818	82.7	2785	81.7
Total	**8512**	**83.3**	**8101**	**79.2**	**7987**	**78.1**
Fireability Queries						
ReachabilityFireability	2567	75.3	2484	72.9	2513	73.7
CTLFireability	2047	60.1	1878	55.1	1695	49.7
LTLFireability	2798	82.1	2639	77.4	2520	73.9
Total	**7412**	**72.5**	**7001**	**68.5**	**6728**	**65.8**
Total query answers	**15924**	**77.9**	**15102**	**73.9**	**14715**	**72.0**

We see that using the method A+B to unfold nets allows us to answer more queries in every category due to the generally smaller nets it unfolds to. In total, we are able to answer 4 percentage points more queries using the unfolded nets of method A+B compared to using the unfolded nets of MCC and 5.9 percentage points more compared to ITS-Tools.

6. Conclusion

We presented two complementary methods for reducing the unfolding size of colored Petri nets (CPN). Both methods are proved correct and implemented in an open-source verification engine of the tool TAPAAL. Experimental results show a significant improvement in the size of unfolded nets, compared to state-of-the-art tools, without compromising the unfolding speed. The actual verification on the models and queries from the 2021 Model Checking Contest shows that our unfolding technique allows us to solve 4% more queries compared to the second best competing tool. In future work, we plan to combine our approach with structural reduction techniques applied directly to the colored nets.

References

[1] Petri C. Kommunikation mit Automaten. Ph.D. thesis, Institut für instrumentelle Mathematik, Bonn, 1962.

[2] Murata T. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 1989. **77**(4):541–580. doi:10.1109/5.24143.

[3] Jensen K. Coloured Petri Nets and the Invariant-Method. *Theoretical Computer Science*, 1981. **14**:317–336. doi:10.1016/0304-3975(81)90049-9.

[4] Dal Zilio S. MCC: A Tool for Unfolding Colored Petri Nets in PNML Format. In: Application and Theory of Petri Nets and Concurrency. Springer International Publishing, Cham, 2020 pp. 426–435. doi: 10.1007/978-3-030-51831-8_23.

[5] Liu F, Heiner M, Yang M. An efficient method for unfolding colored Petri nets. In: Proceedings of the 2012 Winter Simulation Conference (WSC). ISBN 978-1-4673-4779-2, 2012 pp. 1–12. doi:10.1109/WSC.2012.6465203.

[6] Schwarick M, Rohr C, Liu F, Assaf G, Chodak J, Heiner M. Efficient Unfolding of Coloured Petri Nets Using Interval Decision Diagrams, pp. 324–344. Application and Theory of Petri Nets and Concurrency. ISBN 978-3-030-51830-1, 2020. doi:10.1007/978-3-030-51831-8_16.

[7] Hillah L. Family Reunion. https://mcc.lip6.fr/pdf/FamilyReunion-form.pdf, 2021.

[8] Ciaghi A, Weldemariam K, Villafiorita A, Kessler F. Law Modeling with Ontological Support and BPMN: a Case Study. *In CYBERLAWS 2011, The Second International Conference on Technical and Legal Aspects of the e-Society*, 2011. pp. 29–34.

[9] Hillah L. A hot drink vending machine. https://mcc.lip6.fr/pdf/DrinkVendingMachine-form.pdf, 2021.

[10] Muschevici R, Proença J, Clarke D. Modular Modelling of Software Product Lines with Feature Nets. In: Software Engineering and Formal Methods. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011 pp. 318–333. doi:10.1007/978-3-642-24690-6_22.

[11] Kordon F, Bouvier P, Garavel H, Hillah LM, Hulin-Hubard F, Amat N, Amparore E, Berthomieu B, Biswal S, Donatelli D, Galla F, , Dal Zilio S, Jensen P, He C, Le Botlan D, Li S, , Srba J, Thierry-Mieg, Walner A, Wolf K. Complete Results for the 2021 Edition of the Model Checking Contest. http://mcc.lip6.fr/2021/results.php, 2021.

[12] David A, Jacobsen L, Jacobsen M, Jørgensen K, Møller M, Srba J. TAPAAL 2.0: Integrated Development Environment for Timed-Arc Petri Nets. In: Proceedings of the 18th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'12), volume 7214 of *LNCS*. Springer-Verlag, 2012 pp. 492–497. doi:10.1007/978-3-642-28756-5_36.

[13] Jensen J, Nielsen T, Oestergaard L, Srba J. TAPAAL and Reachability Analysis of P/T Nets. *LNCS Transactions on Petri Nets and Other Models of Concurrency (ToPNoC)*, 2016. **9930**:307–318. doi: 10.1007/978-3-662-53401-4_16.

[14] Heiner M, Herajy M, Liu F, Rohr C, Schwarick M. Snoopy – A Unifying Petri Net Tool. In: Application and Theory of Petri Nets. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-642-31131-4, 2012 pp. 398–407. doi:10.1007/978-3-642-31131-4_22.

[15] Heiner M, Rohr C, Schwarick M. MARCIE - Model Checking and Reachability Analysis Done Efficiently. In: Application and Theory of Petri Nets and Concurrency. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013 pp. 389–399. doi:10.1007/978-3-642-38697-8_21.

[16] Chodak J, Heiner M. Spike - Reproducible Simulation Experiments with Configuration File Branching. In: Computational Methods in Systems Biology. Springer International Publishing, Cham. ISBN 978-3-030-31304-3, 2019 pp. 315–321. doi:10.1007/978-3-030-31304-3_19.

[17] Berthomieu B, Ribet PO, Vernadat F. The tool TINA — construction of abstract state spaces for Petri nets and time Petri nets. *International Journal of Production Research*, 2004. **42**:2741–2756. doi:10.1080/00207540412331312688.

[18] Wolf K. Petri Net Model Checking with LoLA 2. In: Application and Theory of Petri Nets and Concurrency. Springer International Publishing, Cham. ISBN 978-3-319-91268-4, 2018 pp. 351–362. doi: 10.1007/978-3-319-91268-4_18.

[19] Amparore EG, Balbo G, Beccuti M, Donatelli S, Franceschinis G. 30 Years of GreatSPN. In: Fiondella L, Puliafito A (eds.), Principles of Performance and Reliability Modeling and Evaluation: Essays in Honor of Kishor Trivedi on his 70th Birthday. Springer International Publishing, Cham. ISBN 978-3-319-30599-8, 2016 pp. 227–254. doi:10.1007/978-3-319-30599-8_9.

[20] Thierry-Mieg Y. Symbolic Model-Checking Using ITS-Tools. In: Tools and Algorithms for the Construction and Analysis of Systems. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-662-46681-0, 2015 pp. 231–237. doi:10.1007/978-3-662-46681-0_20.

[21] Thierry-Mieg Y. Personal Correspondence with Y. Thierry-Mieg, 2021.

[22] Thierry-Mieg Y, Dutheillet C, Mounier I. Automatic Symmetry Detection in Well-Formed Nets. In: van der Aalst W, Best E (eds.), Applications and Theory of Petri Nets 2003. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-540-44919-5, 2003 pp. 82–101.

[23] Klostergaard A. Efficient Unfolding and Approximation of Colored Petri Nets with Inhibitor Arcs. Master's thesis, Department of Computer Science, Aalborg University, 2018. URL `https://projekter.aau.dk/projekter/files/281079031/main.pdf`.

[24] Bilgram A, Jensen P, Pedersen T, Srba J, Taankvist P. Improvements in Unfolding of Colored Petri Nets. In: Proceedings of the 15th International Conference on Reachability Problems (RP'21), volume 13035 of *LNCS*. Springer-Verlag, 2021 pp. 69–84. doi:10.1007/978-3-030-89716-1_5.

[25] Jensen K, Kristensen LM. Coloured Petri Nets, Modelling and Validation of Concurrent Systems. Springer-Verlag Berlin Heidelberg, 1 edition, 2009. doi:10.1007/b95112.

[26] Jensen K. Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use. Volume 1, volume 1. Springer-Verlag Berlin Heidelberg, 2 edition, 1996. doi:10.1007/978-3-662-03241-1.

[27] Christensen N, Glavind M, Schmid S, Srba J. Latte: Improving the Latency of Transiently Consistent Network Update Schedules. *SIGMETRICS Perform. Eval. Rev.*, 2021. **48**(3):14–26. doi:10.1145/3453953.3453957.

[28] Bilgram A, Peter G J, Pedersen T, Srba J, Taankvist PH. Repeatability package for: Methods for Efficient Unfolding of Colored Petri Nets, 2022. URL `https://doi.org/10.5281/zenodo.6417272`.